EMBEDDED LINUX CONFERENCE

No, It's (Still) Never Too Late to Upstream Your Legacy Linux Based Platforms

Neil Armstrong - Linaro - @superna9999@social.linux.pizza

Linaro

EOSS EMBEDDED OPEN SOURCE SUMMIT

Introduction

- Qualcomm Ecosystem Team @ Linaro
  - Qualcomm upstream maintenance
  - U-Boot Qualcomm baseport co-maintainer
  - Bringup/addition of new platforms
- I also maintain other upstream pieces
  - Amlogic SoCs
    - Linux & U-Boot architecture
    - Clocks
    - Pinctrl, Serial, CEC ...
  - DRM
    - Bridge drivers
    - Panel drivers
    - Amlogic Display driver
- Primarily focussed on Linux Kernel
  - 1228 patches in mainline from v3.1 to v6.9-rc1, 240 Qualcomm related
- But also in U-Boot: 270 patches in mainline as v2024.04

Linaro is the software engine of the Arm Ecosystem

Linaro empowers rapid product deployment within the dynamic Arm Ecosystem.

- Our cutting-edge solutions, services and collaborative platforms facilitate the swift development, testing, and delivery of Arm-based innovations, enabling businesses to stay ahead in today's competitive technology landscape.
- Our expertise and contributions spread from Testing & LTS, Security, Cloud & Edge Computing, IoT, AI, CI/CD, Toolchain and Virtualization to vertical projects like Windows on Arm and Android Ecosystem enabling and maintenance.
- Linaro fosters and environment of collaboration, standardization and optimization among businesses and open source ecosystems to accelerate the deployment of Arm-based products and technologies along with representing a pivotal role in open source discovery and adoption.

Linaro has enabled trust, quality and collaboration since 2010

Previous Talk

I already spoke about this almost 8 years ago!

Previous Talk

I already spoke about this almost 8 years ago!

Video:
https://youtu.be/rDMOEvfaMRk

Slides:
https://elinux.org/images/8/8d/Neil_Armstrong.pdf

Warning: Most of the content originates from the 2016 Talk

But I've updated it the current situation and highlighted the notable changes !

Agenda

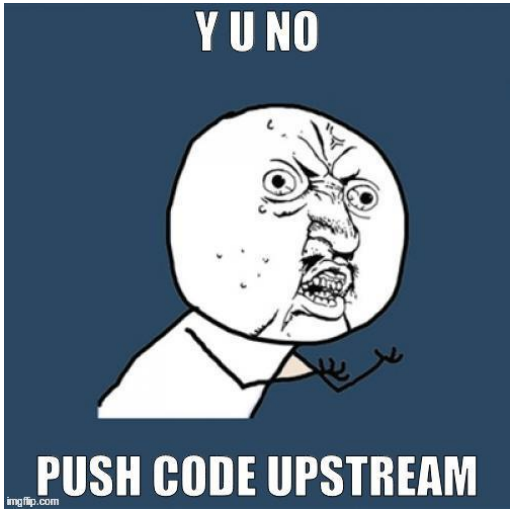
Questions and Answers

- Why Upstreaming ?
- Why mainline kernel is important ?
- Why push code ?
- Which workflow ?
- How long does it take ?
- How hard it is for an “old” platform ?
- What are the benefits ?

Why upstreaming ?

Yeah, why, we have all features working on our v3.12.123 fork with 26345 changes on top, we need to build in an Ubuntu 8.04 container, but it's fine, Qemu will be there when x86_32 gets dropped anyway...

Why upstreaming ?



Always the same question, always the same answers.

More and more vendors had understood the strategic advantages to push code upstream.

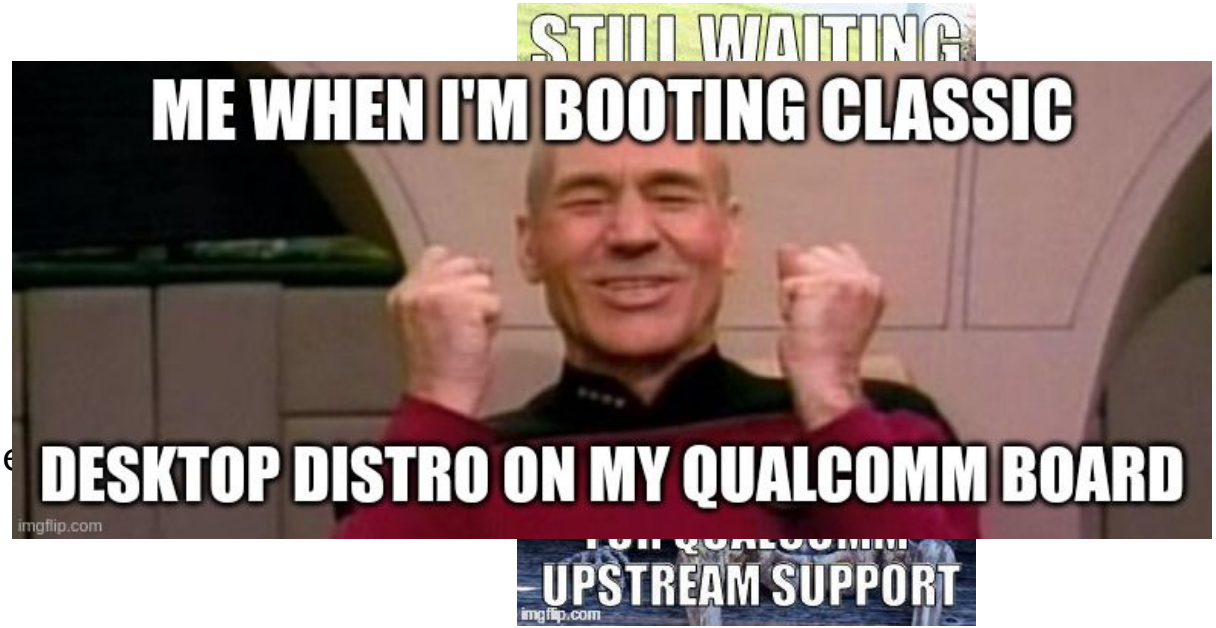
But still very large vendors does not understand :

- They should publish the kernel code (sigh)
- They should push their kernel code in a git repo
- They should push clean code
- They should rework and push code upstream
- They should have a upstream based workflow for their linux products

Who upstreams ?

Hopefully, we can count some vendors that really participate in the upstream work like :

- Intel
- AMD
- Nvidia
- **Qualcomm**
- IBM
- Broadcom
- ARM
- Imagination Technologie
- Realtek
- Marvell
- ST Microelectronics
- ...



Top Contri

Intel
(Unkr
Linar
Googl
Red H
(None
AMD
Pengu
SUSE
Meta
Oracl
NVID
Qualc
Huaw
IBM
Collab
Broad
Arm
Bootli
Renes

Most active 6.8 developers

By changesets

Uwe Kleine-König	368	2.6%
Kent Overstreet	317	2.2%
Lucas De Marchi	189	1.3%
Krzysztof Kozlowski	182	1.3%
Dmitry Baryshkov	148	1.0%
Matt Roper	135	0.9%
Andy Shevchenko	133	0.9%
Andrii Nakryiko	129	0.9%
Matthew Brost	115	0.8%
Matthew Wilcox	113	0.8%
David Howells	108	0.7%
Arnd Bergmann	104	0.7%
Matthew Auld	102	0.7%
Randy Dunlap	102	0.7%
Jakub Kicinski	94	0.7%
Neil Armstrong	90	0.6%
Alexander Viro	90	0.6%
Thomas Zimmermann	83	0.6%
Christoph Hellwig	80	0.6%
Konrad Dybcio	79	0.5%

By changed lines

Arnd Bergmann	59205	7.3%
Matthew Brost	46142	5.7%
Jakub Kicinski	37553	4.6%
Sarah Walker	29771	3.7%
Neil Armstrong	21336	2.6%
Rajendra Nayak	16235	2.0%
Thomas Zimmermann	14881	1.8%
Andrii Nakryiko	12938	1.6%
Kent Overstreet	12617	1.6%
Darrick J. Wong	12403	1.5%
David Howells	10224	1.3%
Nas Chung	10207	1.3%
Ping-Ke Shih	8007	1.0%
Shinas Rasheed	8006	1.0%
Dmitry Safonov	7938	1.0%
Lucas De Marchi	7324	0.9%
Vlastimil Babka	5377	0.7%
Peter Griffin	5263	0.7%
Donald Robson	4911	0.6%
Dmitry Baryshkov	4873	0.6%

What kind of companies does upstreaming ?

We can them separate in those categories :

- SoC vendors
  - Intel, AMD, Nvidia, Qualcomm, Huawei, ARM, Renesas, Realtek, Marvell, ST, ...
- IP Vendors
  - ARM, Imagination Technologies
- Services companies
  - Linaro, Pengutronix, BayLibre, Collabora, Bootlin, Ideas On Board ...
- ODMs,Board/Modules Makers
- Distribution/Product vendors
  - Red Hat, SUSE, Oracle, Google, META, ...

SoC Vendors

- For SoC vendors, Semiconductor industry is harsh, and design costs are very high
  - in addition to various royalties from IP vendors (ARM , Synopsys, ...).
- But some SoC vendors participate actively to make (part) of their SoC family work (partly) upstream.
  - STMicroelectronics is a very good example!
- But these BSPs are often kept on old kernel releases
  - The situation quite improved, we see less v3.x kernels nowadays...
- Some vendors even synchronize regularly with the latest Stable kernel and rebase their BSP on it (Renesas, TI, ST, Qualcomm...)
  - It's probable Google's GKI approach helped here
- Open Source strategy of each SoC vendor should become the priority in the SoC selection for a new product design.

IP Vendors

Usually IP vendors don't really care about having mainline Linux drivers for their IPs, because :

- Most of the time customer aren't using Linux
- Most of the time the IPs aren't Linux accessible
  - Driven by Firmware
  - Hidden behind custom logic
- They sell drivers packages, so why upstream ?

But it's not always true, Cadence has recently started upstreaming drivers for their DSI bridge, and Synopsys engineers does maintain some IP drivers (USB and Ethernet).

ODM, Board/Module Makers

For Board makers, the situation is even more complex.

- Board makers are provided (when possible) with a very custom and complex BSP or SDK which targets either all possible uses cases of the SoC and very specific Reference Design uses cases.
- Yes, sometimes ODMs does not even provide the kernel source (sigh) and even nothing when Android is involved in the product, because the Board vendor will only need to develop its custom Android App.

Hmm, why Upstream kernel is so important after all ?

Yeah I mean we’re fine with our 13y old kernel with all those CVEs and security holes

Mainline Benefits

Relying on Upstream kernel is important because (at least) you can have:

- new features
  - Network features are the most important
  - USB, PCIe, Graphics Drivers
  - Performance Enhancements
  - ...
- bug fixes
- security fixes
- enhanced security features

All of this improves the stability of your product


I'VE FINALLY FOUND IT... AFTER 15 YEARS
THE SCROLL OF TRUTH!
Just upstream code
NYEHHH
imgflip.com

Making a product work is complex enough, why push code upstream ?

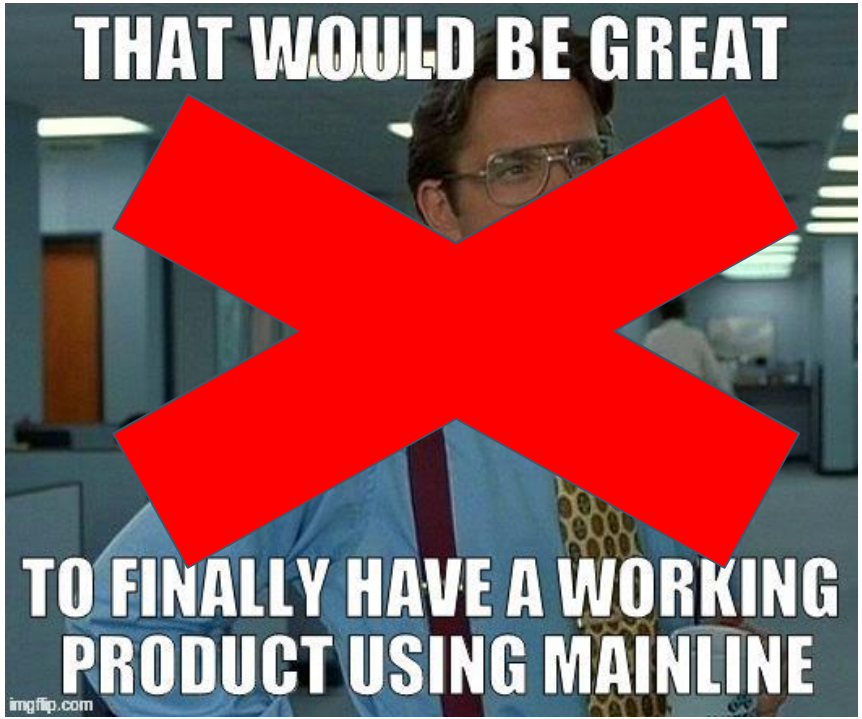
Yeah I mean, we spent 15 years stabilizing our product on the v2.6.32 kernel

Reasons for upstreaming

For obvious reasons :

- Code maintenance ease
- Fair return to the community
- Strengthen the Linux Platform

It’s all !



Code maintenance ease

- Be part of the kernel evolution !
  - code will stay clean and get API changes !
  - Board/SoC support will follow each linux releases.
- when a kernel update is needed, forward porting will be faster and can be done regularly.
- A proper CI/test suite would even accelerate the process
  - Being proactive reduces even more the kernel update process

→ Gain time, gain money !

How should I organize my upstream workflow ?

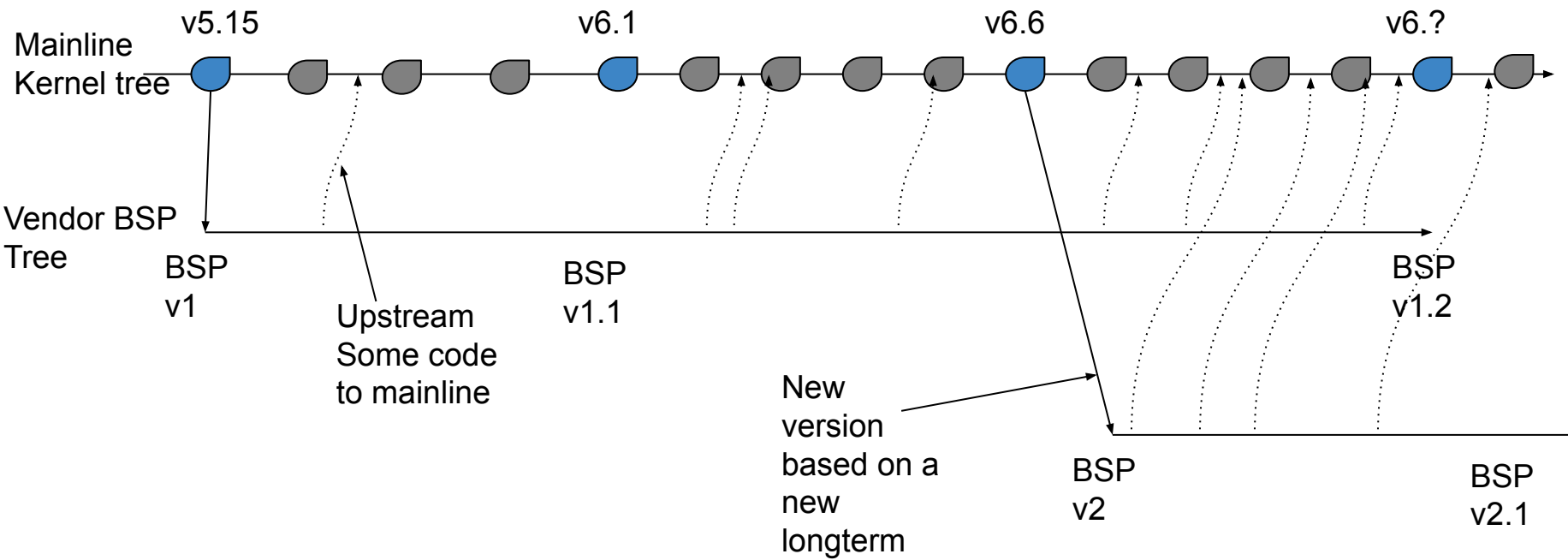
Send all at once on the lists and forget ? Right ?

Workflow examples

Some workflow examples :

- Maintain separate trees
  - one for the BSP and upstream to mainline for client that require recent linux versions
- Port all code from a stable BSP,
  - when nearly complete rebase the BSP on the new linux version
- Iterate by porting the BSP kernel on each long term version
  - meanwhile upstream as much as possible, ...

Classic Workflow



Classic Workflow: Pro / Cons

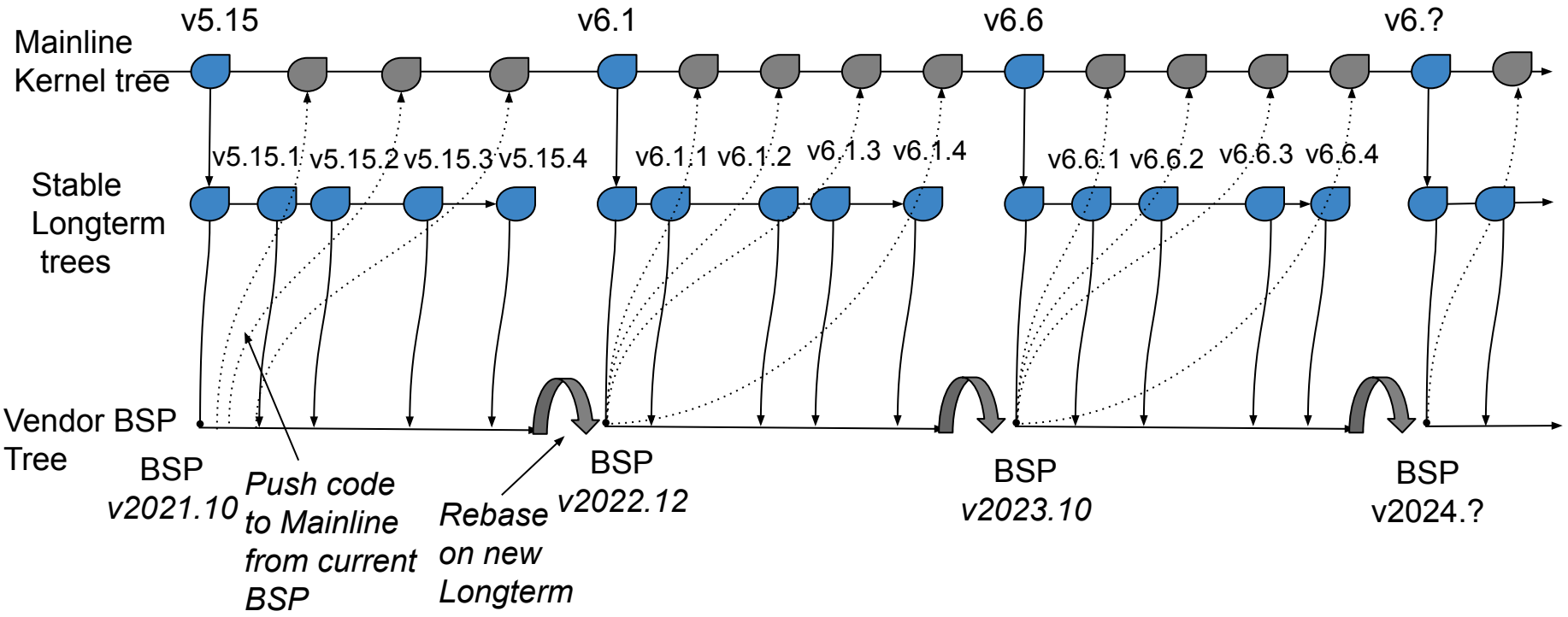
Pros :

- BSP is stable
- Mainline kernel will sometime good support
- Some clients can use mainline

Cons :

- BSP kernel version becomes old at some time
- Rebasing on a new longterm will need a lot of work
- Back porting bugs and security fixes will become harder
- Back porting new features will create a hard to maintain kernel

Mainline aligned workflow



Mainline aligned workflow: Pro / Cons

Pros :

- BSP has always longterm new features, ~each year
- Mainline kernel will get near ~100% support
- BSP driver are reworked and cleaned up

Cons :

- Must maintain a separate team for upstreaming
- Rebasing on each longterm can be complex since drivers has changed
- 1y BSP update may be short for long term supported products, old BSP versions should also have bug and security fixes

How long will it take me to push all this code upstream ?

A few days right ? The intern can definitely do it in his spare time.

Time to Merge

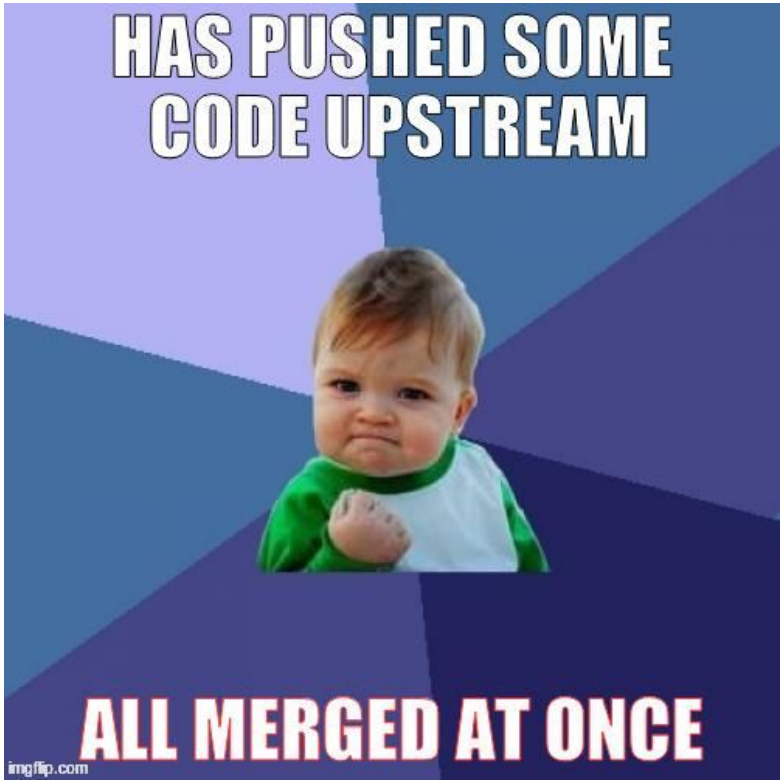
- This a complex question without any clear answers.
- The Linux development cycle must be understood !
- Rework/Refactor is mandatory before and while submission retries.
  - This makes the process really hard to plan
- Depending on subsystems and complexity, submission time can take most of the time.

Mainlining Workflow

- The general mainlining workflow is to push code in each linux subsystems, one by one.
- Coherency of the support for a platform is done over the time.
  - Immediate atomic coherency is hard
  - Simpler to push individual drivers, and then push DT only when every driver has been merged and works
- There is no current “methodology” to push an overall platform support at once, each maintainers will want to have control and review the code to conform to their habits and ease their future work.

Time to mainline

This is why it can take over 2 or 3 releases to get a complete set of patches upstream and have a working kernel on a platform.
The initial support of a platform is the most frustrating period since it will need the full patchset to boot correctly, but for some reasons one subsystem will fail to merge on time for some reasons.



Merge delays

Some of the reasons for the delays are generally :

- Code does not match the subsystem style / code design / architecture (use of deprecated API, ...)
- Bindings are missing / work / untested
  - My colleague Krzysztof Kozlowski speaks about that at 10am today!
- Code depends on headers part of an higher level subsystem (for examples dt-bindings includes for Device Tree support)
- Code depends on partly merged framework API
  - It happens less now, but adding drivers in a middle of a rework needs a special effort
- Patch is posted too late, too close to the next merge window
  - Time is needed for reviewers, especially Dt Bindings since there's a lot to review

Example of delays

Here are some very approximate times :

- Push initial support of a SoC : 2 or 3 versions → ~6 months
- For a simple driver, like PWM, I2C bus, ... :
  - 1 week refactoring / cleanup / patchset preparation
  - 1 day to 1 week for each repost depending on complexity
  - 1 or 2 days for testing before and after merge window
- For a more complex driver like DRM, SATA or Audio driver
  - Initial refactoring time can be much longer, up to 1 or 2 months
  - Time for repost depends on testing time and size of refactoring
  - Such driver upstreaming could be iterated over multiple versions

Global times estimations

- For a simple headless SoC with any power management :
  - **6 months to 9 months**
  - Full time for 1 person, multiple resources won't speed up but will permit more drivers submitted / refactored / tested per version
- For a very complex SoC with Video, Power Management, DSP Audio, modem, ...
  - **18 months to multiple years** depending of the original code quality and SoC complexity (Bus scaling, complex RPC code for Co-processors, complex Audio, secret GPU code, ...)

Practical Example

- Qualcomm Landing team have been upstreaming for the last 10 years
  - Took time to make sure each functionality is correctly designed
  - Added missing support in frameworks
  - Added new frameworks (Ex: Modem, Interconnect, ...)

Result:

- Upstreaming of 90% the Snapdragon 8 Gen 3 took only 1 cycle!
  - Patches were posted the day after the announcement
  - At the end of v6.7 cycle, some patches were able to go in v6,7
- 90% of the patches were merged for v6.8
  - Was available in mainline kernel 3 days after Samsung Phone announcement!
- New features were developed in the meantime
- 90% of these new features + the missing 10% were merged for v6.9

I have an old Linux port for my SoC, how hard it would be to upstream this ?

diff -ur linux.bsp-rev-3.234_final master | mail linux-kernel@vger.kernel.org
Done boss !

But, will it be hard ?

Since the Device Tree migration, the non-x86 support has changed a lot and simplified (is this the right word ?) support for complex SoCs by introducing some frameworks like :

- common clock framework
- pinctrl and gpiod
- generic interrupt
- reset
- drm
- ASoC
- ...

And now a proper way to validate Device Tree with yaml based binding.

But, will it be hard ?

Yes, even for trained professionals

But you can have help, by using :

- Trainings (Linux Foundation, Bootlin, ...)
- Linux Experts (Linaro, BayLibre, Bootlin, Pengutronix, ...)
- Working with the community as Greg KH explains

You can speak with developers and maintainers on IRC, some communities have Matrix servers like PostMarketOS developers.

What about the benefits ?

My boss needs $$$$

Benefits

- Increase in Code quality
  - External review can make the code better
- Minimize Code maintenance cost
  - At least for far less than off-tree
- Enable faster rebase and testing effort
- Clear and Open Software Strategy
  - No more obscure and non-reviewed dirty code
- Customer fidelity over well maintain codebase
- A good way to promote the company within the technical sphere
  - Could also make talented developers work for you

Benefits

- Money

Yes, it will reduce long term R&D costs !

Thank you

Slides?

https://bit.ly/43Eh1Fy

Visit www.linaro.org

Reach out to me at neil.armstrong@linaro.org
or *narmstrong* on Libera.Chat & OFTC
or @superna9999@social.linux.pizza

EMBEDDED LINUX CONFERENCE

Linaro

EOSS EMBEDDED OPEN SOURCE SUMMIT